Sustained Software for Cyberinfrastructure – Analyses of Successful Efforts with a Focus on NSF-funded Software

Craig A. Stewart
Indiana University Pervasive Technology Institute
stewart@iu.edu

William K. Barnett
IUPTI
wkbarnett@iu.edu

Eric A. Wernert
IUPTI
ewernert@iu.edu

Julie A. Wernert
IUPTI
jwernert@iu.edu

Von Welch
Center for Applied Cybersecurity Research, IUPTI
vwelch@iu.edu

Richard Knepper
IUPTI
rknepper@iu.edu

ABSTRACT
Reliable software that provides needed functionality is clearly essential for an effective distributed cyberinfrastructure (CI) that supports comprehensive, balanced, and flexible distributed CI. Effective distributed cyberinfrastructure, in turn, supports science and engineering applications. The purpose of this study was to understand what factors lead to software projects being well sustained over the long run, focusing on software created with funding from the US National Science Foundation (NSF) and/or used by researchers funded by the NSF. We surveyed NSF-funded researchers and performed in-depth studies of software projects that have been sustained over many years. Successful projects generally used open-source software licenses and employed good software engineering practices and test practices. However, many projects that have not been well sustained over time also met these criteria. The features that stood out about successful projects included deeply committed leadership and some sort of user forum or conference at least annually. In some cases, software project leaders have employed multiple financial strategies over the course of a decades-old software project. Such well-sustained software is used in major distributed CI projects that support thousands of users, and this software is critical to the operation of major distributed CI facilities in the US. The findings of our study identify some characteristics of software that is relevant to the NSF-supported research community, and that has been sustained over many years.

Categories and Subject Descriptors
D.2.0 [**Software Engineering**]

General Terms
Management, Economics, Reliability, Experimentation

Keywords
Cyberinfrastructure software; sustainability; reusability; software sustainability; open-source business models

SCREAM'15, June 16, 2015, Portland, Oregon, USA.

ACM 978-1-4503-3566-9/15/06…$15.00.
http://dx.doi.org/10.1145/2753524.2753533

1. INTRODUCTION
Cyberinfrastructure may be defined as consisting of "computational systems, data and information management, advanced instruments, visualization environments, and people, all linked together by software and advanced networks to improve scholarly productivity and enable knowledge breakthroughs and discoveries not otherwise possible" [1]. Software is a core element of cyberinfrastructure and clearly essential for an effective distributed cyberinfrastructure that supports science and engineering research. However, cyberinfrastructure software can be an area of challenge for open (nonclassified) research. A 2011 National Science Foundation task force report included a finding about software, as follows [2]:

> *The current state of cyberinfrastructure software and current levels of expert support for use of cyberinfrastructure create barriers in use of the many and varied campus and national cyberinfrastructure facilities. These barriers prevent the US open science and engineering research community from using the existing, open US cyberinfrastructure as effectively and efficiently as possible.*

The report goes on to highlight key shortcomings in the state of US cyberinfrastructure software as of 2011, when that report was completed, including limitations in software robustness, lack of features, and lack of support.

The National Science Foundation (NSF) supports the creation of many open-source scientific software packages. Perusal of NSF solicitations will reveal more opportunities to fund the creation of new software than to maintain existing software, consistent with the elements of the NSF mission that focus on innovation. Several NSF solicitations for the creation of new software call for including a model for sustainability after the end of NSF funding. The licensing of software created with NSF funds as open source is often a requirement of NSF grant awards. This approach works well in terms of maintaining availability of software and a cost that lowers barriers to adoption.

Our initial hypothesis was that the use of good software engineering methodologies would play a strong role in the sustainability of software, and that objective metrics such as NASA's Reuse Readiness Levels [3] would be informative in distinguishing software that was successfully sustained over a long period of time.

Business models for open-source software are often challenging to implement. This is demonstrated in part by the fate of open-source software studied in a 2007 analysis of open-source business models, many of which have been discontinued since the publication of that study [4]. We became interested in cyberinfrastructure software sustainability as creators, implementers, and supporters of such software. Our interest was driven in part by a desire for sustainability of our own activities and software developed by the Indiana University Pervasive Technology Institute.

Our goal in studying sustainability in cyberinfrastructure was to discover answers to the following questions, particularly as regards software that has received NSF funding or is used by researchers who have received NSF funding:

- *What software testing and hardening methodologies have proved effective in enabling software to be widely used by the NSF research community?*
- *What are best practices in building and testing that contribute to software that is perceived to be highly useful and reliable to the NSF open research community?*
- *What administrative structures, governance processes, and operational infrastructure enable software to be sustainable in the long term?*
- *What financial models provide for long-term sustainability of software as well as supported infrastructure?*
- *What community engagement activities (e.g. requirements gathering, prototyping) demonstrate relevance to the software's eventual success?*

Katz and Proctor [5] recently developed a framework for understanding e-Research Infrastructure, which includes axes of temporal duration, scale (extent of adoption), and purpose (specific within a discipline or general across multiple disciplines). We were specifically interested in middleware and software frameworks. That is, software that is very general in its purpose because of its role as part of distributed cyberinfrastructure. Our primary interests were temporal sustainability, in part because software tends to last longer than hardware as noted by Katz & Proctor, and thus temporal stability is critical to creation of effective distributed cyberinfrastructure.

We adopted a two-stage approach to studying software sustainability. We began with a survey of the community of NSF-funded researchers to determine what they thought were the characteristics of well-sustained cyberinfrastructure software. We went on to conduct a series of in-depth case studies of particular well-sustained cyberinfrastructure software.

The projects selected for case studies were drawn in part from the results of this survey, and in part from our knowledge of interesting projects related to cyberinfrastructure that were operating with potentially interesting business models. In our case studies we focused on software projects that were successful, believing we could learn more from a few model success stories than from many examples of software that had not been sustained. We also thought it difficult for us, as members of the scientific community, to ask our intellectual collaborators and competitors to explain why projects had been discontinued when that question could easily be interpreted as "Whose fault was it that your project lost funding?" We did, however, draw comparisons with some of our own projects that have not turned out to be well sustained and our own informal observations of other unsuccessful software efforts.

Here we present our final analysis and conclusions from this work carried out in 2013 and 2014, supplemented by observations on two projects (Globus Online and Kuali) that have expanded or changed sustainability models since the initial case studies.

Given that sustainability strategies are heavily affected by national funding strategies, our focus in particular was on software that is important to the US open (nonclassified) research community. From the perspective of federal funding agencies in the US, research on software sustainability should inform federal strategies for enhancing US research productivity, innovation, and global competitiveness. In this report, we are particularly concerned with scientific and cyberinfrastructure software in the sense of middleware and software frameworks, as this is the area of concern identified in 2011 by a task force of the National Science Foundation Advisory Committee for Cyberinfrastructure (ACCI). We are not focused on operating systems, or attempting to study the business model of Linux, which is now responsible for billions of dollars of commerce per year. Our goal in preparing this report is to provide insights based on practical experience that will aid research communities in developing software for distributed cyberinfrastructure software and in ways that will enable it to be effective and well sustained over time. Such insights should aid software developers in creating distributed cyberinfrastructure that aids the research community generally in advancing and applying knowledge engineering and science.

2. SURVEY OF NSF-FUNDED PRINCIPAL INVESTIGATORS

Of all US federal agencies that support research, the National Science Foundation has the broadest and most general mandate to support open research. To understand the views and needs of the community of researchers supported by the NSF, we conducted surveys of Principal Investigators funded by the NSF, asking their views on cyberinfrastructure software. We surveyed a random sample of 5,000 individuals drawn from a list of 34,901 individuals who were funded by the NSF in the five-year period 2007-2011. Names were obtained from the NSF publicly available award search feature [6]. The data from the survey are available online at [7]. The Indiana University Center for Survey Research (CSR) administered the study to ensure confidentiality of the data. A total of 685 individuals, or 17% of the people invited, completed the full survey. Most questions were a standard 1-5 Likert scale, with 1 being generally unimportant or bad, and 5 being generally important or good.

2.1 Results of Survey

Respondents. Most respondents identified their primary role as "software user" rather than "software developer" or "other technical role." Thus, this survey reflects the views primarily of the potential adopters of new CI software.

Criteria used by scientists in selecting software. The first survey question asked what factors were important to researchers in selecting a software package. Respondents were asked to rate importance on a scale of 1 (not at all important) to 5 (extremely important). The most important factors – as judged by the average

scores – were as follows (means shown ± 95% confidence intervals):

1) Capabilities and features of a software product are the most important factors to consider when adopting a software package, with a mean score of 4.54 (± 0.05). Respondents overwhelmingly (94%) reported identifying this factor as "important" or "very important."
2) Total cost of ownership: 4.22 (±0.06)
3) Long-term availability: 4.18 (±0.06)
4) Reliability/maturity: 4.16 (±0.05)
5) Initial purchase cost: 4.00 (±0.06)

What criteria make software sustainable, and what software meets those criteria? When asked to evaluate the factors that made a software product sustainable, responses contrasted to those required for adoption. Cited most often were compatibility, availability of support resources, and an active development process. Only 18% of respondents identified software capabilities as key to sustainability. Cost factors ranked near the bottom.

Asked to identify products that met these sustainability requirements, a majority of respondents cited commercial products. The 10 most frequently identified, well-sustained software products were as follows (open-source products are marked with an asterisk):

1) MATLAB
2) Microsoft Office
3) R-project*
4) TeX & La TeX*
5) Mathematica
6) SPSS
7) Adobe Acrobat
8) Linux*
9) Python*
10) EndNote.

Of the top 50 most-cited, commercial products were mentioned roughly twice as often as their open-source counterparts. The most-cited open-source projects include R, TeX/LaTeX, Linux, and Python.

Governance models. Respondents were asked to consider the relative success of some common governance models in open software initiatives in creating an environment for long-term sustainability. There was no clear single frontrunner. The five most frequently indicated items, ranked by average importance score in a range of 1 to 5, were:

1) Hybrid license (commercial/noncommercial users pay different prices) – 441 responses, 3.78 (±0.10) mean score
2) Contributed effort, organizationally supported model (often a corporation supporting an open-source software tool) – 422 responses, 3.65 (±0.10) mean score
3) Meritocracy/volunteer-driven model – 388 responses, 3.41 (±0.11) mean score
4) Membership/foundation model – 355 responses, 3.35 (±0.12) mean score
5) Benevolent/enlightened dictator model – 417 responses, 3.29 (±0.12) mean score.

When asked to cite examples of open software products (or associated companies/consortia/organizations) with governance models that aid the sustainability of their software products, respondents cited a wide range of products with varying governance models. The top tools identified were:

1) Linux
2) R-project
3) Apache
4) Mozilla
5) TeX & LaTeX
6) Python
7) GNU
8) Eclipse
9) OpenOffice
10) ImageJ

2.2 Discussion of Survey Results

The survey response rate was quite reasonable, particularly for an unsolicited email survey with no incentive offered to the potential respondents. Conclusions drawn from the survey represent the opinion of researchers recently funded by the NSF as principal investigators, often leading labs and research groups.

Whether or not a software product was available under an open-source license *per se* was far less a concern for most respondents than were its capabilities, cost, and reliability. However, three of the top five most important selection criteria identified relate to characteristics of open source software – total cost of ownership, long-term availability, and initial purchase cost.

One important piece of information from this survey has to do with the way researchers think about the phrase "cyberinfrastructure software." The letter sent to the 5,000 randomly selected NSF-funded PIs began with the following:

> *I am writing to ask for your help in a landmark NSF study that is being conducted by Indiana University to identify best practices in the development, deployment, and support of robust cyberinfrastructure software.*

When researchers funded by the Division of Advanced Cyberinfrastructure (part of the NSF Computer and Information Science and Engineering Directorate) speak of cyberinfrastructure software, our consistent experience is that cyberinfrastructure software is taken to be middleware and software frameworks – infrastructure building blocks such as Globus Online [8, 9] and software frameworks such as science gateways [10]. The bulk of the respondents seemed to interpret cyberinfrastructure software as end-user applications with scientific uses, such as MATLAB, Microsoft Office, or SPSS. One very strong conclusion from the survey results is that NSF-funded PIs, in general, have a different interpretation of the term cyberinfrastructure software than the substantially smaller community that develops cyberinfrastructure software under funding from CISE. All of the software packages identified as well sustained were viewed as serving purposes sufficiently general that the software was changed over time in response to changing needs, rather than becoming obsolete.

The results of queries about governance models resulted in a set of five most-highly-rated models, all of which are now successfully in use.

This survey helped us identify features that researchers used to select software for their own research, and gather opinions about governance models. It was somewhat less useful in addressing our initial purpose for performing the survey, largely because the way we intended the term cyberinfrastructure software to be used differed from the way many of the respondents understood it. This survey was thus less useful in identifying characteristics of success for the sort of middleware software that was identified as a challenge in the 2011 NSF ACCI Campus Bridging Taskforce. A more extensive analysis of the survey responses is available in [6].

Table 1. Software project case studies – basic info

Project	Description	Year Founded	Primary funding
Primarily end-user applications			
VTK, ITK, CMake, ParaView	Four different but related visualization applications	1998	Initially US Air Force SBIR, now NIH, NSF, DOD, DOE
R project[1]	Statistics and analytics software	1993	Mix of grants, donations, related commercial entities
Galaxy	Scientific workflow system, focused on bioinformatics	2005	NSF, NIH
LAPACK	High performance linear algebra solvers	1985	NSF, DOE, DARPA
Unidata	Data services, tools, support, and training for atmospheric sciences	1982	NSF
Primarily middleware (may deliver end-user applications)			
HUBzero	Web-based scientific workflows (started as NanoHUB)	1995	NSF initially, now contracts and foundation memberships
Kuali	University mission-critical processes (human resources, research administration)	2004	University members of Kuali Foundation
Globus Online	Fast and secure file transfer, data discovery, other cloud services	1997	NIH, NSF, DOE, subscriptions from cloud service users
HTCondor	Distributed high throughput computing	1985	NSF

3. CASE STUDIES

3.1 Case Study Selection of Projects and Methods

We found our survey results less informative than we had hoped in providing information about opinions regarding cyberinfrastructure software in the sense of middleware and software frameworks. We thus selected for detailed case studies a set of software projects that had been well sustained over a number of years or decades, some of which had been ranked highly in the survey responses. Other projects were selected based on our understanding of US software efforts and our initial interests in cyberinfrastructure in the sense of middleware and software frameworks. We attempted to include some diversity in governance models among the projects studied. For example, R was included because it appeared as one of the 10 most widely used open-source tools on our initial survey. HUBZero and Globus Online are included as exemplars of successful software projects, because of their level of adoption in the community and success in creating sustained financial models from a start with NSF funding. Kuali was included because of interest in its business model and its origins without NSF funding. Kitware products were included because of the novelty of their hybrid public / private sustainability model. In our view, all projects qualify as well sustained because they have been in existence for years to decades, and have thousands to hundreds of thousands of users. Unidata was included as an example of a very long-lived CI project. The projects selected for detailed case study were split roughly evenly between middleware and end-user application software. The software projects we studied are listed in Table 1.

We began our case studies with structured questions, then continued with unstructured discussion led by the leaders of software projects. The case studies are thus essentially self-reports from a number of highly successful software projects. We present the key results of case studies as summaries of answers to some of the most critical questions asked of each project in Table 2, shown on the next page.

3.2 Case Studies Results

One aspect of studying software that has been successfully sustained over many years is that the software projects all have relatively general and flexible capabilities. Such software must either have a sufficiently broad suite of functionality that it remains relevant over a long period of time, can be adapted to changing scientific community needs over time, or both.

3.2.1 What licensing terms are preferred?

The software projects we studied involved software released under open-source licenses that are permissive in terms of reuse, most allowing re-use for commercial purposes. Use of an open-source license *per se* was not a distinguishing factor. The many open-source software packages available from SourceForge [11, 12] demonstrate that an open-source license does not automatically mean success. And interestingly, some representatives of successfully sustained projects argued against open source as a basic principle for sustained CI software. In particular, interviewed representatives of some projects described the importance of keeping some core components of a distributed cyberinfrastructure software package held under some other sort of license as a way to ensure the overall sustainability of a software project. For example, the core software that operates tools available from globus.org is not open source, even though the endpoint software is. Similarly, the R project encompasses software such as R Studio, which is licensed, not open source [13].

3.2.2 What administrative structures, governance processes, and operational infrastructure enable software to be sustainable in the long term?

The strongest signal from all our data was that strong, committed, visionary leadership is central to the development of sustained software initiatives. A common characteristic of the projects we studied was the presence of a trusted cohort of operational leadership supporting the vision of one or a small group of project leaders. A geographically centralized core leadership team was another common characteristic.

[1] Information on R was gathered from R project web pages[!]

Table 2. Case studies – governance & user information

Project	Governance	# Users	Annual user meeting?
Primarily end user applications			
VTK, ITK, CMake, ParaView (Kitware)	Company leads, with stakeholder and advisory committee input on the four products	>100,000	Yes
R project	Foundation with governing board, hosted	>1,000,000	Yes
Galaxy	Two PIs as leads (one biologist, one computer scientist)	30,000	Yes
LAPACK	3-person mgmt. team – PI and Co-PIs	Tens of thousands	Yes
Unidata	Director with strategic advisory committee and user committee	55,000	Yes
Primarily middleware (may deliver end-user applications)			
HUBzero	Foundation with board oversees; CEO provides operational leadership	>1,000,000	Yes
Kuali	Foundation with governing board	> 140 higher ed institutions	Yes
Globus Online	University of Chicago owns non-profit company led by PI, with advisory	> 14,000 registered users	Yes
HTCondor	Principal Investigator, with input from key stakeholders	> 100,000	Yes

Leaders of well-sustained projects strongly preferred a permissive open-source license, but at the same time wanted to control official code releases. The community view of any software product depends on how well it functions. Project leaders who control the official code release are adamant that their software, and thus reputation and future funding success, will not be hindered by badly functioning code. (A permissive open-source license means that if bifurcations occur in community interests or among software stakeholders, a code tree can be forked and multiple interests pursued, as has been the case with BLAST [14].

All projects studied have access to good facilities for testing software, high-quality web pages, and high-quality online documentation or help, including web-based self-help. Many provided a mechanism for web-based, community-mediated assistance. This was a factor that was common across software that can be viewed as end-user software or software that provides frameworks accessed directly by end-users – e.g., Kitware products, R, Galaxy, LAPACK, and Unidata, as well as more middleware-oriented software.

3.2.3 What community engagement activities factor into the software's success?

Very proactive plans for engagement within the community of software producers and software users stood out as a strong feature of all software projects we studied. Three factors stood out in particular across most or all of the software projects we studied in depth:

- **Developers engaged with users.** A centralized development team with regular contact with users and leadership was an essential in well-sustained projects. Developers were integrated into support mechanisms, responding to help-desk inquiries; monitoring and participating in listserv, wiki, or blog discussions; and presenting workshops or training classes.
- **Domain experts engaged with developers.** Software initiatives that pair subject-matter (domain) and technological and engineering expertise have an increased probability of broad adoption and sustainability. They are more agile in adapting to changing domain needs, technologies, and trends, and their robust yet flexible products can be expanded or modified, and possibly adopted outside the original project or domain.
- **Conferences or user meetings.** All projects we studied in depth hold an annual conference or user meeting and other regular systematic opportunities for interaction between the software producers and the user community. These interactions among project leaders, developers, and users seem important to the continuing evolution of software to meet user needs.

Conferences and user meetings seem to be particularly important for software that has a strong end-user component, and that is focused on end-users, such as R and Galaxy.

3.2.4 What financial models provide for long-term sustainability and well-supported infrastructure?

The leaders of most projects we studied benefitted from some amount of NSF funding. They see NSF investments as essential in "critical-path" scientific software. These leaders particularly emphasized the value of NSF funding for the utilities, end-user applications, and/or middleware essential to research, discovery, and innovation as a key component of the national scientific cyberinfrastructure. They also pointed out that much CI software has a lifecycle many times longer than that of the individual hardware systems on which the software runs – an observation also made recently by Katz and Proctor [5]. Software project leaders we interviewed generally agreed that it should be viewed and funded as "infrastructure," as critical to research, discovery, and innovation as the more visible NSF investments in supercomputers.

The sustainability plans for Globus Online, HUBzero, and HTCondor all stem in part from use levels in the hundreds of thousands of users. Globus Online and HUBzero have found ways to turn utilization into cash flow. Several of the projects, such as HTCondor and Globus Online, weave together funding from multiple federal agencies as part of their sustainability strategy. Galaxy has fewer users than some of the other projects studied. However, the needs it meets and level of adoption by the scientific community have generated funding and allowed for sustained growth in its capabilities and user base.

Kitware is singular among the projects we studied. This novel public/private partnership stands out for the extent to which a private-sector company fosters the sustainability of open-source software. The partnership depends so heavily on the expertise of the private company leaders that it seems not easily replicable. However, the idea of a company started with an SBIR (Small Business Innovation Research) grant award, and operating largely as a government contractor, seems efficient for the Federal Government and useful for the software developers and communities served.

Unidata was also distinctive among the projects we studied in having a sustainability strategy based on ongoing NSF funding. For Unidata this has been and likely can remain a viable financial strategy because their core value proposition is the distribution of data and tools to analyze those data – and the data are critical to science and national civil security. The atmospheric data and tools to analyze those data are essential to many areas of science supported by the NSF. These data are central to our understanding of atmospheric processes and the environment in general. They are also very important in disaster prediction and response related to severe weather conditions.

Kuali was founded with a grant from a private foundation, and has never received direct NSF funding. With its focus on university financial and business processes, it is an outlier in our study. What is notable is that in the face of a significant shared challenge – the cost of commercial enterprise software – universities and colleges were able to band together and create commercial-quality software for highly-regulated enterprise activities. Indiana University is a participant in the Kuali project. As of October 2013, Kuali had saved Indiana University some $20 million [15].

3.3 Recent Changes in Software Projects

Since we completed these case studies there have been interesting adaptations in the business models of two of the projects we studied – Globus Online and Kuali. Globus Online is pursuing a model that may prove very interesting: The Internet2 NET+ initiative, described as follows [16]:

> *Through Internet2 NET+, members collectively identify and vet community and industry cloud solutions that are (or could be) effective in meeting campus challenges, and have the potential to scale to benefit all member institutions' teaching, learning and research needs. …Those services that pass the rigorous evaluation are made available to all member institutions. … and aggregating the demand of the Internet2 membership provides the best possible standard pricing and terms.*

NET+ represents a fundamental shift in supporting activities of the US research community. In the past, the NSF often paid institutions to offer services to the US research community. In the NET+ model, the US research community pays directly for services offered to support its activities. As need for computational resources expands, and the NSF budget expands little if at all, the NET+ model may benefit the US research community and US global competitiveness. One obstacle to adopting NET+ tools is that most services require InCommon membership, a hindrance for smaller schools. So far, Globus Online remains in the "service validation" phase of Internet2 NET+.

Kuali has completely changed its business model since the initial case studies were completed. Recognizing deficiencies in its service model and in its ability to sustain itself in the long term, the Kuali Foundation in summer 2014 created and invested heavily in a new and completely independent for-profit, professional open-source company, KualiCo. The Kuali Foundation retains majority control of the new company, and is the company's largest investor. It maintains significant oversight of how the company evolves, and ensures that it maintains its focus on higher education needs and earns profits in a way congruent with higher education ideals. The Kuali Foundation has the right to designate a director on the KualiCo Board of Directors and has an "exceptional" veto right to prevent the sale of the company, an IPO of the company, or a change to the open-source license agreement. This was designed to move Kuali from a provider of products serving only a handful of institutions to an entity that serves a full suite of products (including financial, human resources, student information, research, and library systems for higher education) for a large number of US universities and colleges, thus creating a more sustainable financial model with development costs distributed more evenly across more institutions. A portion of KualiCo's profits will be returned to the Kuali Foundation, which will now turn some of its resources to exploring other unmet needs in the higher education market sector. In this regard, Kuali is maintaining higher education control and open-source licenses, but trying to capture a larger market share via a commercial entity. Kuali is an outlier among the software products we studied because it began without NSF funding, and because its business model is based largely on US universities' desire to avoid the cost of commercially-developed solutions for mission-critical activities such as human resources and grants management.

3.4 Comparison with Earlier Research on Open-source Software

There have been a number of analyses of open-source software methodologies and economies, including books by Eric Raymond [17] and compendia of research such as [18]. Major open source projects are described online, e.g.: the Open Source Initiative (OSI) [19] and the Apache Foundation [20]. Our work looks at a particular segment of the open-source community – cyberinfrastructure software with an emphasis on projects that have benefitted from NSF funding. We do not believe that anything we have discovered contradicts other current research on best practices for creation and sustainability of open-source software. All projects we studied have some sensible form of open-source or community license, use good coding practices, and are coded in modern software languages. Some have lasted long enough to have undergone a complete rewrite. Furthermore, our characterization of licensing and governance models, while not identical, is generally consistent with that of Katz and Proctor [5].

Within the niche of software funded in part by the NSF or used by NSF-funded researchers, we have identified characteristics that several well-sustained projects have in common:

- A utility and/or flexibility of use sufficiently broad that the software remains relevant over many years
- A strong core of committed leaders, most often co-located in one geographic area
- Control over the definitive software versions and effective test and build processes
- Deep and effective engagement with users
- Use of business models that evolve over time.

There is some indication that governance models may also evolve over time. For example, R started out as a project of two faculty members, and is now operated by a strong community led by a board of directors. While we did not do an in-depth case study of iRods (Integrated Rule-Oriented Data System) [21] we note with respect and admiration that Reagan Moore devoted a career to developing this software, which now seems to have such a solid group of leaders and strong user base that it will continue to be widely used long after Dr. Moore's retirement.

Comparing lists of characteristics with software projects in which the authors were directly involved, and that were not successfully sustained, the most common distinction was a narrow scope of utility. We and our collaborators have created software that was useful for a while, and then came to an end because the project

leaders or funding agencies found the software no longer worthy of investment of time, money, or both. That said, even when individual project leaders lose interest and/or funding and stop leading a particular software project, it is possible with open-source software for the core functionality to be picked up and maintained by others. Such is the case with one particular set of functionalities that were successfully maintained over decades, but under different project leaders and with different software names. The function in question is maximum likelihood inference of phylogenetic trees, sustained over time by four different project leaders. The first package in this family tree was Felsenstein's DNAml [22] through two versions of a package called fastDNAml – projects led by Olsen [23] and then Stewart [24]. This same functionality is now available – along with many new additions and optimizations – in Stamatakis' package RAxML [25]. This basic functionality was sufficiently useful that the core functionality of maximum likelihood analysis was maintained and expanded over more than 30 years, even as particular leaders picked up work on the tool, and then went on to other things as their interests changed.

4. BENEFITS OF WELL-SUSTAINED SOFTWARE

In this section we address some of the benefits of well-sustained cyberinfrastructure to the scientific communities supported by the US National Science Foundation and US taxpayers — the ultimate source of funds for government-supported research.

4.1 Open-source Software & Scientific Reproducibility

Multiple articles, including Stodden [26] stress the importance of reproducibility in research that makes use of scientific computing tools. Use of open-source software enhances the chances that a given analysis or cyberinfrastructure experiment can be replicated. A former colleague, the late Richard Repasky, made a tar ball (tar archive, or archive of files made with the Unix tar utility [27]) each time he completed a research project. It contained the distribution of Linux, R, and all software in the computing environment of the system he used for the analysis; the make scripts used to install the system; his data sets; his R scripts; and the output of his analyses. Theoretically, as long as someone has access to enough software to unpack a tar archive and an x86 emulator, it should be possible to replicate and expand the analysis of that data on into the future. Such an analysis has a straightforward path to reproducibility in, say, 50 years. How one finds a valid license key for today's version of commercial software such as SPSS or SAS in 50 years is less certain. This example is focused on R – primarily an end-user application – but the same principle holds true for scientific libraries and LAPACK as compared to NAG, and for the ability to replicate any type of computer analysis done with open-source software.

4.2 Well-sustained Software and Cybersecurity

The scientific / engineering community strives to produce perfect software that functions flawlessly. If the software is used in certain contexts, these flaws become vulnerabilities eroding the cybersecurity of the systems in which they are deployed. These flaws pose a barrier to sustainability of developing and releasing corrective patches. Last year's examples, e.g. Heartbleed and Shellshock, show that even in software that has been around for a number of years, new vulnerabilities may still be found. In an environment of changing cybersecurity threats, sustainability becomes complicated. It requires clear policies on what software is supported and will be patched, technical processes for producing and distributing patches, and processes for communicating to the community of software users.

Efforts are underway to reduce the frequency of security flaws in open-source cyberinfrastructure software by integrating continuous software testing with the software development process, so-called continuous integration or, when addressing vulnerabilities, continuous assurance. Examples include Jenkins [28] and BaTLab [29] for continuous integration, and the Software Assurance Marketplace (SWAMP) [30] for continuous assurance. A strong argument can be made for continuous and ongoing investment in open-source cyberinfrastructure so as to enable the ongoing maintenance of expertise in the code base and enable updates and enhancements to ensure security of software in the face of ever-changing threats and attacks on software.

4.3 Examples of Well-sustained Cyberinfrastructure & Large-scale, Distributed CI Implementations: OSG and XSEDE

The Open Science Grid [31] operates the largest distributed high-throughput computing (HTC) facility in the world, based on reusing and / or adapting existing open-source software [32,33]. OSG was immensely successful in its initial purpose of analyzing data from the Large Hadron Collider. OSG has packaged HTCondor and associated software such that many campuses could add their own resources to the Open Science Grid, or create their own Virtual Organizations that operate within the Open Science Grid. An example of the OSG scale of resources is the fact that during a 12-month period prior to the submission of this paper, it completed more than 200,000,000 computer jobs, consuming more than 800,000,000 CPU hours. OSG will create its own software when needed, but as a matter of organizational strategy, does that only when necessary.

XSEDE, the eXtreme Science and Engineering Discovery Environment [34] is a major NSF-funded organization supporting the NSF XD (eXtreme Digital) program cyberinfrastructure resources and one of the largest US distributed cyberinfrastructure facilities. Through digital services and support it supports more than a dozen supercomputers, storage systems, and advanced visualization systems throughout the US [35-37]. XSEDE focuses on hardening and implementing pre-existing software, and does little software implementation of its own. In other words, an aggregate investment by the NSF of well over $100M in CI system deployment used by the US research community is operated with software that comprises largely noncommercial, open-source software. The importance of maintaining software that is integral and essential to the operation of such large distributed cyberinfrastructure efforts is self-evident.

4.4 Well-sustained CI Software and Research Education

For decades, researchers carefully managed laboratory notebooks. Today, knowing how to use research software is as basic to research as good laboratory data management practices. The ability to conduct good software management practices is a prerequisite for developing sustained software and making research replicable. Well-sustained open-source software provides students with resources for learning the craft of research using the same software used by leading US scientists. It enables such programs as the Software Carpentry project [38], which teaches

basic Unix and software engineering tool skills. With a decreased emphasis on such material in computer science classes, such programs create otherwise hard-to-find and important training tools for the scientists and engineers of tomorrow.

4.5 But is it Transformative?

In an effort to encourage a focus on cutting-edge ideas, the NSF review criteria for grant proposal solicitations often include the potential transformative impact of the proposed research. The NSF explains "transformative research" as follows: "Transformative research involves ideas, discoveries, or tools that radically change our understanding of an important existing scientific or engineering concept or educational practice or leads to the creation of a new paradigm or field of science, engineering, or education" [39].

That a software package works well one year, and does so three years later, is hardly transformative by this definition. The NSF, however, explains further: "Other potentially transformative research proposals may request support for key incremental or threshold advances (e.g., new methods or analytical techniques) that, if successful, could put a discipline on a new scientific trajectory, provide tools that allow unprecedented insights, or radically accelerate the rate of data collection." Viewed in this light, sustained cyberinfrastructure can enable transformative research. For example, verifying of the existence of the Higgs boson is unquestionably a transformative research outcome. HTCondor and the Open Science Grid enabled the data analysis that confirmed the existence of the Higgs boson.

5. CONCLUSIONS

We analyzed software funded by the National Science Foundation or used by researchers funded by the NSF, using largely social science approaches (surveys and case studies).

A survey sent to NSF-funded principal investigators about software that was sustained revealed several interesting pieces of useful information. First, the word "cyberinfrastructure" is still interpreted differently within the computational science community than by the community of scientists and engineers supported by the NSF as a whole. In a survey on cyberinfrastructure software of NSF-funded PIs across all NSF-supported areas, a very large number of responses included software that the computational science community would more likely consider "end-user applications" than "cyberinfrastructure." Community adoption and use of the term cyberinfrastructure remains a work in progress [10].

The survey of NSF PIs indicated that they select software first on the basis of functionality. The next most important factors, in order of importance, indicated by NSF PIs were:

- Total cost of ownership
- Long-term availability
- Reliability/maturity
- Initial purchase cost

Even though software being open source *per se* was not one of the five most important factors in this survey, these criteria are addressed or met by high-quality, open-source software. Among producers there is a strong interest in open-source software for the sake of allowing community-driven processes and because of NSF guidelines that often specify that software developed with NSF funds be open source. An interesting variant on this opinion came from a small number of software projects we studied that had either "commercial / open-source" variants of the software, or open-source software endpoints as part of a distributed computing system, and maintained rights and control over central organizing software. This latter approach is taken by the University of Chicago Computation Institute with Globus Online tools. This is partly because of the need to maintain control over the code to ensure that the core management portion of the online services functions properly, and partly related to funding and sustainability strategies.

Among the software projects we studied in depth, three factors stood out as distinctive: governance and leadership models, control over the definitive code tree, and community processes. The governance and leadership of most of the successful projects we studied can be described as having a strong core of committed leaders that are geographically co-located. This strong leadership and control over the definitive code tree were common across almost all of the projects we studied that had been well sustained over time. The third factor that stood out among highly successful projects was a systematic mechanism for regular, ongoing interaction between software creators and their user community – including an annual user meeting. Project leaders stated that user meetings in particular enable sharing of expertise, and enable leaders of software projects to understand what new interests and challenges are perceived by their user community.

We found overall that software projects persist in part because they meet an important need, and in part because they are led by a group of leaders who are deeply committed to the continuity of those projects. This dedication to persistence has in some cases led to experimentation and evolution of funding models over time. The Globus Project, for example, has experimented with three different funding models since its inception.

Globus Online and HUBzero demonstrate a shift in funding strategies for delivering services to the national research community. In the past, the NSF or other federal funding agencies have paid an institution (or group of institutions) to deliver services to the national research community without direct cost to the community for using those services. The NSF supercomputer centers of the '80s and '90s, the TeraGrid, and XSEDE are examples. Globus Online and HUBzero are experimenting with models that charge for use of services (though not full chargeback). This reflects a notable and qualitative change in provisioning services to the national community.

There are important scientific and societal benefits to sustaining cyberinfrastructure software projects such as the Open Science Grid and XSEDE. Each has supported research that resulted in a Nobel Prize – criteria for which include changing scientific paradigms and discoveries of great importance to mankind. The Open Science Grid and XSEDE have supported hundreds of other research projects with demonstrated or potential societal benefit, and great inherent scientific value. Sustained cyberinfrastructure software projects are essential to the success of distributed cyberinfrastructure systems. The two largest open (nonclassified) research distributed CI systems in the US – XSEDE and OSG – are dependent upon sustained, open-source software, and could not operate without it. Sustained, open-source software is critical to the cybersecurity of distributed CI systems and to the reproducibility of scientific and engineering analyses.

There are no obvious blueprints for project leaders to follow in enabling a software project to be well sustained over time. Our findings are that open-source licenses, good software engineering practices, and strong testing practices and quality control are necessary but not sufficient conditions for cyberinfrastructure projects to be sustained over time. One common distinguishing factor – dedicated, committed, and visionary leadership – echoes

the traditional wisdom among venture capital firms: that the leadership of an endeavor is more important than anything else. In some cases particularly successful projects have grown from leadership of one or two scientists to be led by a larger group or board of directors.

Strong software quality control and control of the definitive software distribution were also consistent among the well sustained software projects we studied. Another factor that stood out as common to the successful projects we studied was the importance of annual user meetings or conferences. In addition, several of the sustained software projects we studied had a financial model based on a hybrid of grant award funding and commercial services.

There is no doubt about the importance of sustaining cyberinfrastructure software projects to the development and operation of distributed cyberinfrastructure systems, particularly US NSF-funded cyberinfrastructure. The model "federal funding agencies support it year after year" is untenable in general in the foreseeable future. This approach is unsustainable given the rise in demand for cyberinfrastructure software brought about by the growth in creation of digital data combined with the generally stagnant levels of federal support for science and engineering. The growth in cyber attacks makes well-managed software all the more critical to US research endeavors.

Our study focused on a particular segment of the software development community, where most software development efforts are wholly or largely open source and the user community is largely researchers funded by or doing work relevant to the mission of the US National Science Foundation. Our findings are generally consistent with earlier studies of open-source software and add detail as regards this particular community. We hope that these findings are useful to those who develop and use cyberinfrastructure, and that this paper contributes to community understanding of the strategies and tactics required to create an effective distributed cyberinfrastructure supporting scientific innovation and discovery.

6. ACKNOWLEDGMENTS

This research is based upon work supported in part by the National Science Foundation under Award No. 1147606 and builds on work supported by NSF awards 1002526 & 0829462. This work was also supported by the Indiana University Pervasive Technology Institute, which was initiated with major funding from the Lilly Endowment, Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF) or other supporting organizations. We thank the many personnel affiliated with the projects we studied for spending hours with us as part of our case studies. Thanks to anonymous reviewers for very helpful and insightful comments that improved this paper. Thanks to Jan Holloway and Jeremy Fischer for editorial assistance. Thanks to Scott Michael for assistance with statistical analyses.

7. REFERENCES

1. Dreher, P., V. Agarwala, S. Ahalt, G. Almes, S. Fratkin, T. Hauser, J. Odegard, J. Pepin, and C.A. Stewart. 2009. *Developing a Coherent Cyberinfrastructure from Local Campuses to National Facilities: Challenges and Strategies.* Available from: http://hdl.handle.net/2022/5122.
2. NSF Advisory Committee for Cyberinfrastructure Task Force on Campus Bridging. 2011. *Final Report*. Available from: http://www.nsf.gov/cise/aci/taskforces/TaskForceReport_CampusBridging.pdf.
3. NASA Earth Science Data Systems - Software Reuse Working Group. 2010. *Reuse Readiness Levels (RRLs), Version 1.0*. Available from: https://earthdata.nasa.gov/sites/default/files/esdswg/reuse/Resources/rrls/RRLs_v1.0.pdf
4. Chang, V., H. Mills, and S. Newhouse. 2007. *From Open Source to long-term sustainability: Review of Business Models and Case studies*. All Hands Meeting 2007, OMII-UK Workshop 2007; Available from: http://eprints.soton.ac.uk/263925/.
5. Katz, D.S. and D. Proctor. 2014. *A Framework for Discussing e-Research Infrastructure Sustainability*. Available from: http://dx.doi.org/10.5334/jors.av.
6. National Science Foundation. 2015. *Award Search*. Available from: http://www.nsf.gov/awardsearch/.
7. Wernert, J., E Wernert, J. Fischer, H. Terhune, A. Bowers, T. Miller, C.A. Stewart. 2014. *Best Practices and Models for Sustainability for Robust Cyberinfrastructure Software - Survey Dataset and Analyses. Indiana University.* Available from http://hdl.handle.net/2022/17312
8. University of Chicago Computation Institute. 2015. *Home Page*. Available from: https://www.ci.uchicago.edu/.
9. Globus Project. 2015. *Home Page*. Available from: https://www.globus.org/.
10. Stewart, C.A., Richard Knepper, Matthew R. Link, Marlon Pierce, Eric Wernert, Nancy Wilkins-Diehr. 2014. *Cyberinfrastructure, Science Gateways, Campus Bridging, and Cloud Computing.* In: Encyclopedia of Information Science and Technology, Third Edition (10 Volumes) 2014; pp 6562-6572]. Available from: http://hdl.handle.net/2022/18608.
11. Slashdot Media. 2015. *Home page.* Available from: http://slashdotmedia.com/.
12. SourceForge. 2015. *Home page*. Available from: http://sourceforge.net/.
13. R Studio. 2015. *R Studio Pricing*. Available from: http://www.rstudio.com/pricing/.
14. National Center for Biotechnology Information. 2015. *BLAST® Basic Alignment Search Tool*. Available from: http://blast.ncbi.nlm.nih.gov/Blast.cgi.
15. Rivard, R. 2014. *Kuali Tries to Compete* https://www.insidehighered.com/news/2013/10/15/colleges-prepare-major-software-upgrades-kuali-tries-woo-them-corporate-vendors
16. Internet2. 2015. *Internet2 NET+*. Available from: http://www.internet2.edu/netplus/.
17. Raymond, E.S. 2010. *The Cathedral and the Bazaar*. http://www.catb.org/esr/writings/cathedral-bazaar/.
18. Stewart, C.A., Knepper, R. D., Link, M. R., Pierce, M., Wernert, E. A., Wilkins-Diehr, N. 2014. *Cyberinfrastructure, Science Gateways, Campus Bridging, and Cloud Computing.* In*:* Encyclopedia of Information Science and Technology, Third Edition. IGI Global. Hershey. PA. Available from: http://hdl.handle.net/2022/18608
19. Open Source Initiative. 2015. *Home page*. Available from: http://www.opensource.org.

20. Apache Foundation. 2015. *Home Page*. Available from: https://www.apache.org.
21. iRODS. 2015. *Home page.* Available from: https://www.irods.org/.
22. Felsenstein, J., *Evolutionary trees from DNA sequences: a maximum likelihood approach.* J Mol Evol, 1981. 17(6): p. 368-76.
23. Olsen, G. J., H. Matsuda, R. Hagstrom, and R. Overbeek. 1994. *fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood.* Comput. Appl. Biosci. 10: 41-48.
24. Stewart, C.A., D. Hart, D. K. Berry, G. J. Olsen, E. Wernert, W. Fischer. 2001. *Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference.* Proceedings of SC2001, Denver, CO, November 2001. http://portal.acm.org/citation.cfm?id=582054
25. Stamatakis, A. 2014. *RAxML Version 8: A tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies.* Bioinformatics (open access). Available from: doi:10.1093/bioinformatics/btu033.
26. Stodden, V. 2012. *Reproducible research for scientific computing: Tools and strategies for changing the culture.* Computing in Science & Engineering, vol.14, no. 4, pp. 13-17, July/August 2012, doi:10.1109/MCSE.2012.38.
27. The Open Group. *Tar File Archiver*. Available from: http://pubs.opengroup.org/onlinepubs/7990989799/xcu/tar.html.
28. Jenkins. *Home Page*. Available from: http://jenkins-ci.org/.
29. BaTLab. 2015. *Home Page.* Available from: https://www.batlab.org/.
30. *SWAMP (Software Assurance Marketplace).* 2015. *Home page.* Available from: https://continuousassurance.org/.
31. Open Science Grid. 2015. *Home page*. Available from: http://www.opensciencegrid.org/.
32. Pordes, R. 2008. *Challenges Facing Production Grids*. In: High Performance Computing and Grids in Action. Advances in Parallel Computing, 16th ed. IOS Press: Amsterdam. p. 506-521.
33. Pordes, R., W. Kramer, M. Livny, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, R. Quick. 2007. *The Open Science Grid--Its Status and Implementation Architecture*. In: *International Conference on Computing in High Energy and Nuclear Physics (CHEP 07)*. Available from: *http://tinyurl.com/pun3vbj*
34. XSEDE. 2015. *Home page*. Available from: https://www.xsede.org/.
35. XSEDE. 2015. *XSEDE Resources Overview*. Available from: https://www.xsede.org/web/guest/resources.
36. Towns, J., T.Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G.D. Peterson, R. Roskies, J.R. Scott, N. Wilkins-Diehr, *XSEDE: Accelerating Scientific Discovery.* Computing in Science & Engineering. 16:62-74. Available from: doi:10.1109/MCSE.2014.80
37. XSEDE. 2015.*What We Do*. Available from: https://www.xsede.org/what-we-do.
38. Software Carpentry. *Home page*. Available from: http://software-carpentry.org/.
39. National Science Foundation. *Characteristics of Potentially Transformative Research*. 2010 31 Jan 2011]; Available fro*m: http://tinyurl.com/q9s6tln*.

All web citations active as of 20 April 2015.

8. Postal Address of Authors

The postal address of all authors except Von Welch is: Indiana University, 2709 E. 10th Street, Bloomington, IN 47408. The Postal address for Welch is: Indiana University, 2719 E 10th Street, Bloomington, IN 47408.